Dissipative Cellular Automata As Minimalist Distributed Systems: A Study On Emergent Behaviors

Franco Zambonelli[1,*], Andrea Roli[2], Marco Mamei[1]

[1]Dipartimento di Scienze e Metodi dell'Ingegneria – Università di Modena e Reggio Emilia
[2]Dipartimento di Elettronica Informatica e Sistemistica – Università di Bologna

[*]**Contact Author:** franco.zambonelli@unimo.it

Abstract

This paper describes the behavior observed in a class of cellular automata that we have defined as "dissipative", i.e., cellular automata for which the external environment can somehow inject "energy" to dynamically influence the evolution of the automata. In this class of cellular automata, we have observed that stable macro-level global structures emerge from local interactions among cells. Since dissipative cellular automata express characteristics strongly resembling those of open distributed systems, we expect that similar sorts of macro-level behaviors are likely to emerge in real world systems of the same nature and need to be studied, controlled, and possibly fruitfully exploited. A preliminary set of experiments reporting two ways of indirectly controlling the behavior of DCA are reported and discussed w.r.t. the possibility of applying similar sort of indirect control on open distributed systems.

1. Introduction

Engineering distributed systems is notoriously a challenging task. One of the main problems is that designing system's component in isolation, on the basis of the consolidated structured design principles (and thus following a reductionist approach) tend to miss those features arising when the system's components executes concurrently, in a common environment, influencing each other behavior. These system-level features, usually referred as system's *emergent behaviors*, may have a very strong impact on the overall system's behavior. For example, it has been recently discovered that the Internet and the Web have evolved and behave in rather peculiar and unexpected (emergent) way, strongly impacting on performance and reachability of information [AlbJB99].

The analysis of emergent behavior has been almost completely neglected by current best practices in engineering distributed systems, however as the consequence that these systemic behaviors starts to be widely recognized, more and more researches address this topic, trying to exploit these emergent behaviors to engineering and control complex distributed systems [ZamP02]. Works on swarm based systems [Bon99, Par97], amorphous computers [Abe00], cellular automata [Wol94, Bar97], etc. share the idea of studying and engineering system's level (i.e. emergent) behaviors to control distributed systems.

In this context, this paper discusses some experiments that we have performed on a new class of cellular automata that we have defined as Dissipative Cellular Automata (DCA). DCA differ from "traditional" cellular automata [Wol94, Bar97] in two characteristics: while "traditional" cellular automata are composed of cells that interact with each other in a synchronous way and that are influenced in their evolution only by the internal state of the automata themselves, dissipative ones are asynchronous and open. First, cells in the DCA update their status independently of each other, in an "autonomous" way. Second, the automata live dipped in an environment that can directly influence the internal behavior of the automata, as in open systems. We think that openness will be a key issue in next generation distributed applications. The Internet and the upcoming embedded computer networks will soon rend obsolete the idea of designing a system completely isolated from the rest of the world, and environmental influences and openness in general will soon become prime design concerns.

From this point of view, DCA can be considered as a minimalist open distributed system [Jen00, ZamJW01] and, as that, their dynamic behavior is likely to provide

useful insight into the behavior of real-world systems.

DCA exhibit peculiar interesting behavior, as the experiments reported in this paper show. During the dynamic execution of the DCA, stable macro-level spatial structures emerge from local interactions among cells, a behavior that does not emerge when the cellular automaton is synchronous and closed (i.e., when the state of a cell is not influenced by the environment). On this basis, the paper argues that similar sort of macro-level behaviors are likely to emerge in distributed systems populating the Internet and our physical spaces. Such behaviors are likely to dramatically influence the overall behavior of our networks at a very large scale. This may require new models, methodologies, and tools, explicitly taking into account the environmental dynamics, and exploiting it during software design and development either defensively, to control its effects on the system, or constructively, as an additional design dimension.

On this basis, the paper also shortly describes two different methodologies that we have tried to apply in order to indirectly control emergent behaviors in DCA (i.e., to control which pattern, among the several possible ones, to make emerge in a DCA). Finally, the paper discusses how similar sorts of indirect control can possibly applied to open, distributed systems.

2. Cellular Automata

Cellular Automata (CA) are regular lattices of cells, each one being a finite-state automaton. According to simple dynamics, cells update their state depending on a (typically simple) state transition function of their state and of the state of neighboring cells. The scientific interest on CA comes primarily from the fact that, despite the simplicity of local rules, they can show complex global behaviors.

Formally, a CA is statically defined by a quadruple

$$A = (S, d, V, f),$$

where S is the finite set of possible states a cell can assume, d is the dimension of the automaton, V is the neighborhood structure, and f is the local transition rule. The automaton structure is a d-dimensional discrete grid $L=Z^d$, where Z is the set of integers. Each cell is identified with an array of d components $\mathbf{i}=(i_1,...,i_d)\in L$ which represent the coordinates of the cell in the grid. It is generally assumed that the grid is infinite, either not limited or closed to a d-dimensional torus. The state of a cell is expressed as a variable x whose domain is defined by S; and the ordered list of cell states defines the CA global state X. The neighborhood structure V defines which cells "influence" any cell. V is defined as a function $V:L\rightarrow\wp(L)$ which maps a cell to a set of cells. The neighborhood structure is regular and isotropic, i.e., V has the same definition for every cell. Usually, V is a subset of the group of translations in L. Finally, the local transition rule is a function $f:S^V\rightarrow S$ which maps a configuration of states in a neighborhood to a state. The transition rule defines the future state of a cell depending on the state of its neighbors (and, possibly, the state of the cell itself). f is typically the same for each cell (uniform CA).

While the above defined quadruple A specifies the "static" characteristics of an automaton, the complete description of a CA requires the definition of its dynamics, i.e., of the dynamics ruling the update of the state of the CA cells. The usual definition of CA is with synchronous dynamics: cells update their state in parallel at each time step. However, synchronous dynamics is hardly representative of real-world phenomena, making it not suited for the modeling and the simulation of those phenomena involving a population of autonomous interacting elements, for which asynchronous dynamics have to be introduced. In the experiments presented in this paper, CA have an asynchronous dynamics [IngB84, LumN94]: at each time, one cell has a uniform probability of rate λ_a to autonomously wake up and update its state.

The behavior of a CA under synchronous and asynchronous dynamics can be very different [Wol94, Shr99]. In this paper, we consider 2-dimensional CA (NxN square grids with wraparound borders) with two states. These kinds of CA have been deeply studied and have also a biologic interpretation: cells can be interpreted as alive/dead, depending on their state. As an example, let us consider a 2-states CA ($V=\{0,1\}$, where cells are said to be dead or alive, respectively), the Moore neighborhood structure (the neighbors of a cell are the 8 one defining a 3x3 square around the cell itself) and the following transition rule (RULE A):

f = {*a died cell gets alive iff it has 2 neighbors alive; a living cells lives iff it has 1 or 2 neighbors alive*}

Once a synchronous CA starts to evolve from an initial random situation, the states of all cells synchronously change accordingly to the above rule, and after a transient eventually reaches the final cyclic attractor of which one of the composing global states is shown in Figure 1. Figure 2 shows the same CA having evolved accordingly to asynchronous dynamics: the CA usually reaches a fixed-point attractor that its synchronous counterpart has never been observed to be able to reach.

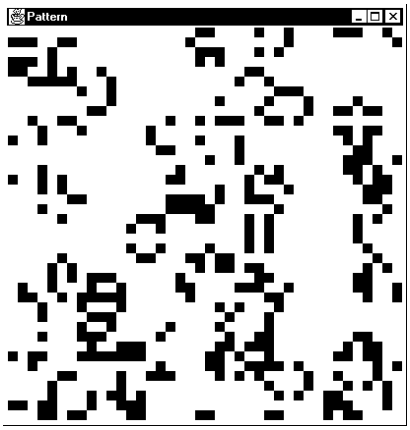

Figure 1: **State of a cyclic attractor in a synchronous CA - RULE A**

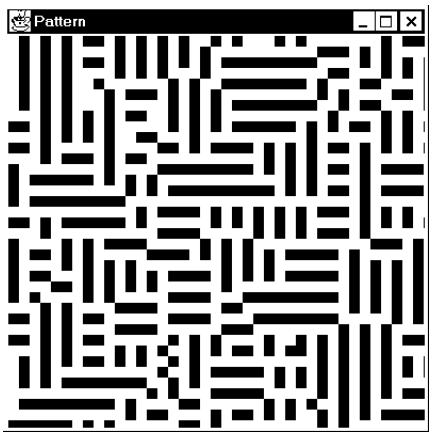

Figure 2: **A fixed point attractor in an asynchronous CA - RULE A**

3. Dissipative Cellular Automata

3.1. Description

CA studied so far are closed systems, as they do not take into account the interaction between the CA and an environment. Instead, the new class of CA that we have studied is, in addition to asynchronous, "open", in the sense that the dynamic behavior of the CA can be influenced by the external environment. From an operative point of view, the openness of the CA implies that some cell can be forced from the external to change its state, independently of the cell having evaluated its state and independently of the transition function (Figure 3).

By considering a thermodynamic perspective, one can consider this manifestation of the external environment in terms of energy flows: forcing a cell to change its state can be considered as a manifestation of energy flowing into the system and influencing it [NicP89]. This similarity with thermodynamic systems made us call this kind of CA as dissipative cellular automata (DCA), in that the DCA consumes external energy to reach a final (regular, as shown in the following) configuration.

From a more formal point of view, a DCA can be considered as:

- $A = (S, d, V, f)$;
- asynchronous dynamics (with uniform distribution of rate λ_a);
- a perturbation action $\varphi(\alpha,D)$.

where A is the quadruple defining a CA, the dynamics is the one already discussed in Subsection 3.1, and the perturbation action φ is a transition function which acts concurrently with f and can change the state of any of the CA cells to a given state α with some probabilistic distribution D, independently of the current state of the cells and of their neighbors. Specifically, in our experiments with $V=\{0,1\}$, $\alpha=1$ and D is a uniform distribution of rate λ_e.

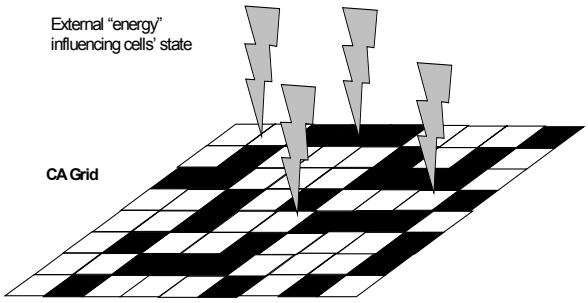

Figure 3: **Dissipative cellular automata**

3.2. Emergent Behavior

The behavior exhibited by DCA is dramatically different from both their synchronous and closed asynchronous counterparts. In general, when the degree of perturbation (determined by λ_e) is high enough to effectively perturb the internal dynamic of the DCA (determined by the rate of cell updates λ_a) but it is still not prevailing over it so as to make the behavior of the DCA almost random (which happens when λ_e is comparable λ_a), peculiar patterns emerge. The interested reader can refer to the page *http://polaris.ing.unimo.it/DCA/* to repeat the experiments on-line.

We have observed that the perturbation on the cells

induced by the external - while keeping the system out of equilibrium and making impossible for it to reach any equilibrium situation - makes the DCA develop large scale regular spatial structures. Such structures exhibit long-range correlation between the state of the cells, emerged despite the strictly local and asynchronous transition rules, and breaks the spatial symmetry of the final state. In addition, such structures are stable, despite the continuous perturbing effects of the external environment.

As an example, Figure 4 shows a pattern emerged from a DCA, both exhibiting stable macro-level spatial structures. For this DCA, the transition rules and the neighborhood structure are the same of the CA described in Section 2: the presence of global-scale patterns - breaking the spatial symmetry of the automata - is evident. As another example, Figure 5 shows a typical pattern emerged for a DCA with a neighborhood structure made up of 12 neighbors (all cells having a maximum distance of 2 from the cell itself) and with the following transition rule (RULE B):

f = {*a died cell gets alive iff it has 6 neighbors alive; a living cells lives iff it has 3,4,5, or 6 neighbors alive*}

Again it is possible to see large-symmetry breaking patterns emerge, extending to a larger scale than the local patterns emerging under asynchronous but closed regime (Figure 6).

The phenomenon underlying the behavior of DCA are very similar of the ones determining the emergence of large-scale structures in dissipative systems [NicP89], e.g., in Bénard's cells, where, the temperature gradient between the two plates is substituted by the ratio λ_e / λ_a. When this ratio is small, expressing small perturbations, each autonomous component (a DCA cell), acting asynchronously accordingly to strictly local rules, tend to reach a local equilibrium (or a strictly local dynamics), which reflects in a global uniform equilibrium of the whole system. When the ratio λ_e / λ_a ratio increases, the system is kept in a substantial out-of-equilibrium situation, resulting in continuous attempt to locally re-establish equilibrium. This typically ends up with cell groups having found new equilibrium states more robust with regard to the perturbation (or compatible with it). Such stable local patterns start soon dominating and influencing the surrounding, in a sort of enforcing feedback, until a globally coordinated (i.e., with large scale spatial patterns) and stable situation emerges. When the degree of perturbation is high enough to avoid local stable situations to persist for enough time, they can no longer influence the whole systems, and the situation becomes turbulent: spatial patterns disappear and the DCA dynamics becomes highly disordered. Detailed quantitative analysis are in progress.

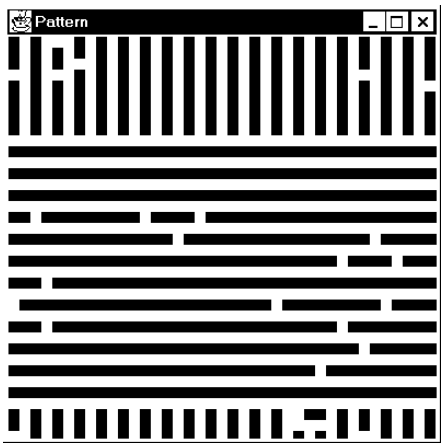

Figure 4: **A Behavior evolved in a DCA - RULE A**

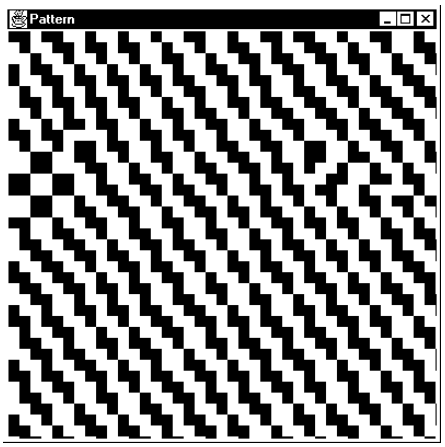

Figure 5: **A behavior evolved in a DCA - RULE B**

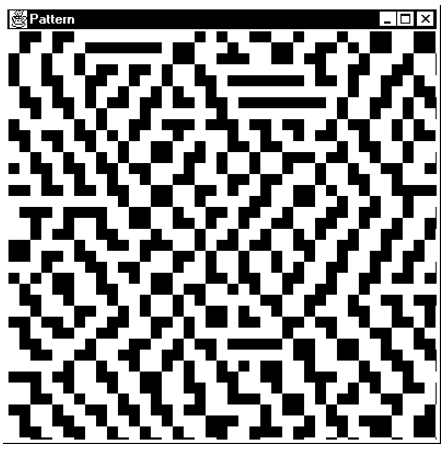

Figure 6: **A stabilized situation in an asynchronous closed CA - RULE B**

4. Open Distributed Systems VS. DCA

There are three characteristics which are typical of open distributed systems (and that are more and more characterizing all types of software systems) that are reflected in DCA: autonomy of components, locality in interactions, situatedness in an open and dynamic environment.

Components of a distributed system are autonomous entities [Jen00], in that their execution is not subject to a global flow of control. Instead, the execution may proceed asynchronously, and the component's state transition occur accordingly to local internal timings. This is actually what happens, because of the adopted dynamics, in DCA: each state transition in a DCA cell is driven by an internal clock, which is independent from the clock - and the state transitions - of the other DCA cells.

For the sake of scalability and efficiency, distributed systems typically execute in a spatially bounded distributed domain, and wide-area - inter-organizations - interactions are limited as much as possible. In DCA, a cell interacts with (that is, can check the state of) only a limited number of other cells in its neighborhood.

Components are situated entities that live dipped in an environment, whether a computational one, e.g., a Web site, or a physical one, e.g., a room or a manufacturing unit to be controlled. The component is typically influenced in its execution (i.e., in its state transitions) by the events happening in the environment. In this sense, distributed systems are "open systems": the global evolution of the system may be influenced by the environment in which it lives [ParBS02]. And, in most of the cases, the environment possesses a dynamics that is not controllable or foreseeable. For instance, the computational resources, the data, the services, as well as the other components to be found on a given Web site cannot be predicted and they are likely to change in time. Analogously, the temperature and lightening condition in a room that a component is devoted to control may vary dynamically for a number of reasons that cannot be predicted. This sort of openness is the same that we can find in DCA, where the perturbation of the environment, changing the internal state of a cell, can make us consider the cell as situated in an environment whose characteristics dynamically change in an unpredictable way.

Given the above similarities, and given that the characteristics leading to the observed DCA behaviors are present in most of today's distributed systems there are very good reason to presume that similar strange behaviors will be observed as soon as these systems will start populating the Internet and our physical environments.

4.1. Defending from Undesired Emergent Behaviors

The reported experiments open up the possibility that a distributed system immersed in a open and dynamic environment may exhibit behaviors very different from the ones it was programmed for. Of course, this is not desirable and may cause highly damaging effects [ParBS02]. For instance, in the case of Internet pricing systems, and despite theoretical equilibrium results, environmental dynamics can make macro-level spatial patterns emerge, leading to great price differences in different sites of the planet. In the case of cooperative distributed information retrieval, this may cause a large amount of available information to be left out from the search, while making the remaining part over-accessed.

Accordingly, we think that a re-thinking of the methodologies currently adopted for the design, development, and maintenance of open, distributed systems is required to avoid such situations to occur, or at least to be able to predict and control them. By now, software systems are designed in a mechanical way, component by component, so as to exhibit a specific, deterministic behavior. Such approach immediately fails when a large number of autonomous components are involved. Modern distributed systems researches recognize this problem, and have gone farther, by approaching the study of engineering such systems in more systemic, macro-level, terms [Jen00]. The next challenge is approaching the study of distributed systems by making the environment and its dynamics play a central role. In other words, one should design a system so as to make it exhibit, under a wide range of environmental conditions, the desired global behavior, disregarding if necessary the full understanding of the behavior of its components, and rather trying to understand the behavior of the system as a whole depending on the environmental conditions. Possibly, a software system should be designed so as to be able to re-adapt itself dynamically and make its internal dynamics contrast the environmental one.

As a consequence of the macro-level approach and of the primary role of environmental dynamics, open and distributed systems will be no longer tested with the goal of finding errors in them (or in their components), but they will be rather tested with regard to their capability of behaving as needed as a whole, independently of the exact behavior of its component [Huh01], and under the environmental conditions in which the system is expected to operate when released. It is also important to note that, in most of the cases, a newly deployed software system will execute in an environment where other systems already executes. Thus, the new software system will impact on the environmental condition of the pre-existing

systems and, by executing, on their environmental dynamics. Thus, designing and testing a system will not only be devoted to make a software system useful, but also to guarantee that it will not be dangerous to other systems.

4.2. Exploiting Useful Emergent Behaviors

Clarifying the dynamic influences between multi-agent systems and their environments can make environmental dynamics become an additional design dimension, rather that an enemy to fight.

As a very trivial application example, directly inspired from the visual appearance of the DCA patterns, one could think at "intelligent paintings". Paintings can be made up of active, radio-enabled, micro-components, able change their colors according to local transition rules, and making it possible to change the color patterns via simple radio-commands perturbing the transition rules and causing a global change in the pattern of a wall. As another example, the possibility of making global patterns emerge from a system relying on local interactions could be exploited so as to enforce global control in a wide-area distributed system with very low efforts.

More generally, one could think at exploiting the environmental dynamics to control and influence a open, distributed system from "outside the loop" [Ten00], that is, without intervening on the system itself. In a world of continuous computations, where decentralized software systems are always running and cannot be stopped (this is already the case for Internet services and for embedded sensors) changing, maintaining and updating systems by stopping and re-installing them is not the best solution, and it could not be always feasible. Instead, given the availability of proper models and tools, one could envision the possibility of influencing the system without stopping it, simply forcing specific environmental dynamics changing the global behavior of the system so as to make it exhibit the required behavior.

5. METHODOLOGIES TO CONTROL DCA BEHAVIOR

Form the previous discussion, it turns out that it would be of fundamental importance to have the possibility of controlling emergent behaviors in DCA. In fact, by considering again the relations with distributed computing, having the possibility of controlling emergent behavior may open up the doors to both defending from the emergence of undesirable behaviors and exploiting useful behaviors by making them emerge as needed.

However, due to the characteristics of such systems (large number of components, large-scale distribution, autonomy of components) one cannot think at imposing behavior via direct control on all its components. Instead, such control must be as much distributed and decentralized as possible, and can rely only on he possibility of controlling a few components of the systems, without making any assumption on the possibility of controlling all components and their dynamic interactions.

To this end, we experienced two complimentary way of controlling DCA behaviors, both having lead to successful, and rather surprising, results.

5.1. Rule-based Control

The rule-based methodology we have experienced amount at changing the rules determining state transitions in cells. With respect to distributed systems, such rule changes would translate in injecting in components of a the system (e.g., via mobile code technologies) some new working parameters and activities. In the DCA, to make a desired pattern emerge, the rule as to change so as to make (some of the) cells recognize that the cells in the neighborhood are in the right configuration, i.e., in a configuration that approaches the configuration that would be required for the required pattern to emerge.

Among a variety of possible rule modifications, we have found that such rule modification should not be too strict, e.g., a cell should lives if and only if its neighborhood is in one of the possible configurations that would have assumed in the presence of the desired pattern. For such strict rules, all cells in the automata quickly die. Instead, we have found out that such rules modifications have to be very weak and, counter-intuitively, should enable a cell to get to life and live in a wider range of configuration than the unmodified rule allow. For instance, given the generic rule:

f = {*a died cell gets alive iff it has between D1 and D2 neighbors alive; a living cells lives iff it has between L1 and L2 neighbors alive*}

The simple modified rule mf that enables a specific pattern to emerge is in the form:

mf = {*rule f OR a cell must live and get alive as soon as the there are alive neighbor cells in one of the correct configuration w.r.t. the desired patterns*}

which can re-phrased as follows: a cell must follows the normal transition rules, however, independently of that, and independently on the number of alive and died cells, a cell must get to life whenever in the neighborhood there are alive cells in the right position.

Applying the mf rule, we have been able to make any

desired pattern emerge from any initial configuration from a DCA, independently of the chosen *f* rule and independently of the dimension of the DCA grid. Of course, such configuration emerges only in the presence of a correct range of value for the λ_e /λ_a ratio. For instance, we have been able to make the pattern of Figure 7 emerge from a DCA by applying the modified rule B, a pattern that emerged only very rarely (about 0,01% of the cases) in previous experiments.

Of course, for a control methodology to be applicable to large distributed systems, it must not assume the capability of influencing the behavior of all the components of the system. For this reason, we have tested the rule-based methodology also by modifying the rule only in a sub-set of the DCA cells. Such experiences have been very satisfying, in that the rule-based enables to make the desired pattern emerge even when only a very low percentage (down to 30%) of the cells applies the modified rules.

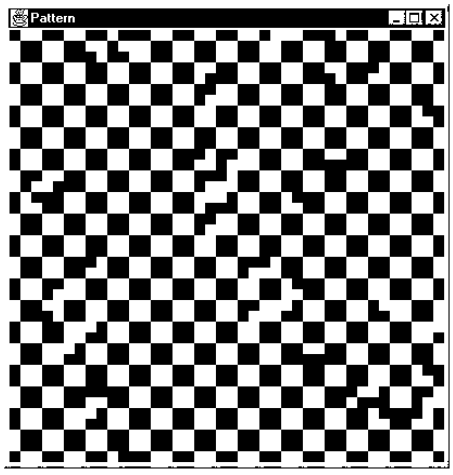

Figure 7. **Rule-based methodology: a rare pattern whose emergence can be controlled**

5.2. Generalization-based Control

The generalization-based methodology starts from totally different considerations, and gets its inspiration from Hopfield's work on neural networks [Hop82]. It is known that Hopfield's networks can generalize a pattern from imposition of a partial pattern. Should DCA exhibit such generalization property, controlling them would imply initializing a localized sub-set of the DCA cells in accord to the desired patterns (see Figure 8), and then let the DCA evolve by making the global pattern spontaneously emerge from the local imposition. In the case of a distributed system, such methodology would require the possibility of controlling the activities of a local cluster of components, and then let this locally imposed control diffuse to the whole system.

At the beginning, we were not really convinced of DCA possessing such generalization property. Instead, rather surprisingly, we found out that, for a few rules we have experienced, such methodology worked well: for a desired global rule to emerge of a 40x40 DCA grid, we had to initialize a local portion of about 20% of the global grid size. In the case of a distributed system, this means for instance that controlling the initial state of cluster of a few hundred components may be enough to influence a distributed system of several thousands.

Further experiments have to be performed to analyze in better detail such generalization properties, and to quantify their capability of working in larger grids and for larger sets of rules.

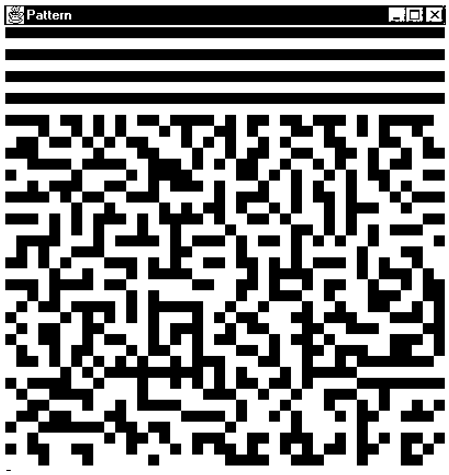

Figure 8. **Generalization-based methodology: a local pattern imposed on a portion of a DCA grid.**

6. CONCLUSIONS AND FUTURE WORKS

Large-scale spatial patterns, not observed under closed regime, emerge in open DCA. Since open distributed systems exhibits all of the characteristics of DCA (autonomy of components and openness) similar sort of structures are likely to make their appearance as soon as large systems of this kind will start populating the Internet. This requires methodologies, and tools, to predict and control emergent behaviors in these systems, and enabling to either exploit emergent behaviors as an additional design dimension, or to prevent undesirable behaviors.

The experiments reported in this paper - and the two methodologies proposed to control emergent behaviors -

are indeed preliminary. Currently, we are trying to better formalize the concepts of "openness" and of "perturbation", and to characterize and measure the degree of perturbation and the degree of order of the emergent patterns. Also, we are trying to define effective models for controlling emergent behaviors in open distributed systems, with a key focus on mobility of components [MamLZ02]. The main goal is to make our experiments more and more approximate the characteristics of real systems in mobile scenarios and, eventually, to end up with a more realistic simulations.

Acknowledgments. Work partially supported by Nokia Research Center Boston and by MIUR Project "MUSIQUE".

References

[Abe00] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman and R. Weiss, "Amorphous Computing", Communications of the ACM, 43(5), May 2000.

[AlbJB99] R. Albert, H. Jeong, A. Barabasi, "Diameter of the World Wide Web", Nature, 401:130-131, 9 Sept. 1999.

[Bar97] Y. Bar-Yam. Dynamics of Complex systems. Addison-Wesley, 1997.

[Bon99] E. Bonabeau, M. Dorigo, G. Theraulaz, "Swarm Intelligence", Oxford University Press, 1999.

[CabLZ02] G. Cabri, L. Leonardi, F. Zambonelli, "Engineering Mobile Agent Applications via Context-Dependent Coordination", IEEE Transactions on Software Engineering, 2002, to appear.

[GusF01] R. Gustavsson, M. Fredriksson, "Coordination and Control in Computational Ecosystems: A Vision of the Future, in Coordination of Internet Agents, A. Omicini al (Eds.), Springer Verlag, pp. 443-469, 2001.

[Hop82] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proceedings of the National Academy of Science USA, 79:2554-2558, 1982.

[IngB84] T. E. Ingerson, R. L. Buvel, "Structure in Asynchronous Cellular Automata", Physica D, 10:59-68, 1984.

[Jen00] N. R. Jennings, "On Agent-Based Software Engineering", Artificial Intelligence, 117(2), 2000.

[LumN94] E. D. Lumer, G. Nicolis, "Synchronous Versus Asynchronous Dynamics in Spatially Distributed Systems", Physica D, 71:440-452, 1994.

[MamLZ02] M. Mamei, L. Leonardi, F. Zambonelli, "Co-Fields: Towards a Unifying Model for Swarm Intelligence", Technical Report No. DISMI-UNIMO-2002-3, University of Modena and Reggio Emilia, February 2002.

[NicP89] G. Nicolis, I. Prigogine, Exploring Complexity: an Introduction, W. H. Freeman (NY), 1989.

[Par97] H.V.D. Parunak, "Go to the Ant: Engineering Principles from Natural Multi-Agent Systems", Annals of Operations Research 75:69-101, 1997.

[ParBS02] V. Parunak, S. Bruekner, J. Sauter, "ERIM's Approach to Fine-Grained Agents", NASA/JPL Workshop on Radical Agent Concepts, Greenbelt (MD), 2002.

[SchR99] B. Schönfisch, A. De Roos, "Synchronous and Asynchronous Updating in Cellular Automata", BioSystems, 51(3):123-143, 1999.

[Sip99] M. Sipper. "The Emergence of Cellular Computing". IEEE Computer, 37(7):18-26, July 1999.

[Ten00] D. Tennenhouse, "Proactive Computing", Communications of the ACM, May 2000.

[Wol94] S. Wolfram. Cellular Automata and Complexity. Addison-Wesley, 1994.

[ZamJW01] F. Zambonelli, N. R. Jennings, M. J. Wooldridge, "Organizational Abstractions for the Analysis and Design of Multi-agent Systems, 1st Workshop on Agent-Oriented Software Engineering, LNCS No. 1957, Jan. 2001.

[ZamP02] F. Zambonelli, V. Parunak, "From Design to Intentions: Sign of a Revolution", 1st International Joint Conference on Autonomous Agents and Multi-agent Systems, Bologna (I), July 2002.